International Journal of Control and Automation
Vol.8, No.8 (2015), pp.189-202
http://dx.doi.org/10.14257/ijca.2015.8.8.20

The Design and Implementation of a Continuous Playback System for Radio Broadcasting *

Yi Guo[1] and Qiong Li[2]

[1]*School of Electronic and Information Engineering, Xihua University, Chengdu, China*
[2]*School of Automation Engineering, UESTC, Chengdu, China*
[1]*lpngy@qq.com,*[2] *liqionger1017@163.com*

Abstract

For a playback system of radio broadcasting, the stability and continuity are very important factors. Once occurred some situations, such as audio file error, network and database connections error, broadcasting system software or hardware error, and the virus infections, the output of a playback system may be discontinuous which will make the audience feel uncomfortable. A continuous playback system was designed and implied in this paper, which can ensure continue broadcasting of the audio file while some error happened. The realization of this method is suggested by DirectShow technology.

Keywords: *Continuous Playback, Seamless Switch, DirectShow, Buffer, Broadcast*

1. Introduction

To achieve digital audio workstation security broadcast, its stability, reliability, security aspects need high requirements, broadcast, to 7 x 24 hours running, is the most basic requirements. In the whole process of broadcasting system, there will be some problems, such as network failure, network server intermittent cannot return data, disk error led to the audio file error.

At this stage, to solve the problems mentioned above, broadcasting system is directly switched to backup broadcast file playback, but sometimes a short pause, or backup file broadcast a file on the same server appears in this stage, when the network disk error[1], will be unable to return data, it will directly to broadcast the error, even not to mention to seamlessly switch. Therefore, this solution can not realize seamless switch on real significance, which is broadcast from the point of view, are not able to achieve seamless switch, the effect of smoothing.

In view of this situation, we adopted the design of multiple files servers, more than one file server backup file repeat broadcast, and adding buffer mechanism when reading the file data, so as to realize seamless handoff in the broadcast effect. Because we made redundant time, enough to buffer audio data, therefore, that broadcast file error time and broadcast file path switch and time required less time redundancy. Broadcast from the point of view, seamless handover broadcast file, strengthen the error-tolerant of the broadcast system, guaranteeing the workstation can run steadily and reliably.

This technology has been applied for Chinese national invention patents, which patent No is 201110144970.5. The method described in this paper has been applied to dozens of radio broadcasting system, and ensuring the reliable broadcast.

ISSN: 2005-4297 IJCA

2. The Seamless Switch Principle and Method

2.1 The Seamless Switch Principle

The seamless switch principle of the audio file is given as follow:

1) Multiple file server design, determining a file server for the current file server, the other is the backup file server. The files of the current file server backup [2] in all other backup server. To provide physical support for multiple paths to read file.

2) The data buffer design, data buffer has enough buffer play data, when the broadcast file error, path switch ensure the length of the data buffer play time is greater than that found the error and the time of path switch.

From the above two points, we can draw the conclusion, principle of broadcast file seamless switch is the first multi path buffer to read file data into the data buffer and then broadcast the data read from the data buffer zone, when an error is discovered and switch, because the data buffer to buffer data enough, the broadcast data will not be interrupted supply, so as to ensure the continuity of the broadcast, the broadcast aspect can achieve a seamless handover broadcast file.

2.2 The Seamless Switch Method

DirectShow technology is aiming at solving the multimedia processing efficient technical solutions based on Microsoft windows platform, through the DirectShow technology platform, designed the seamless switch technology in broadcasting system [3]. The technology has the following several points: multi path to read file; data buffer mechanism; Source Filter based on DirectShow technology (read the file filter).

First, introduced the multi paths to read the file and data buffer mechanism, these two points are inseparable, complementary.

Follow this flow chart, we can clearly find out the work flow of the data buffer model, which was described as bellow.

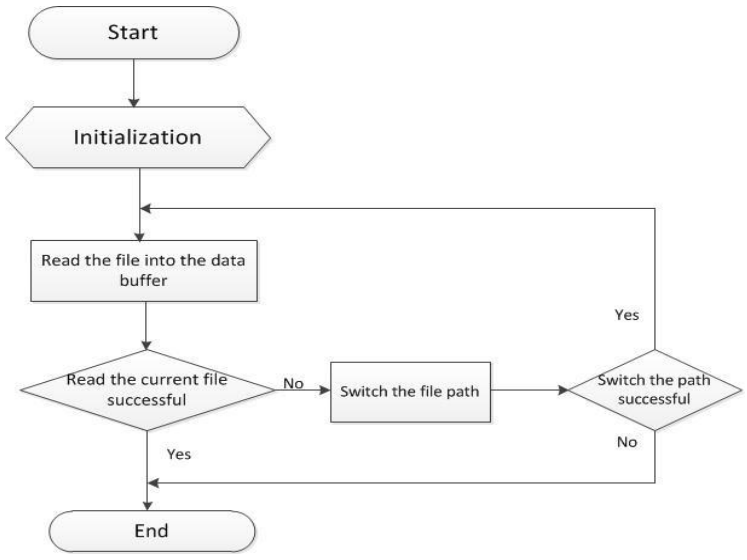

Figure 1. The Flow Chart of Multi Path Buffer to Read File

Start reading the file, an initializing operation mainly initializes multiple files read paths and sets the current path, in addition to initialize the buffer operation, set the buffer memory for the spatial distribution of the initial size. After the initialization operation, according to the current file paths read and write file into data buffer, if read the current file successfully, then the end of the file is read; if the current file read failure or unable

to return data reading process, switching file backup path switching path, if successful, will restore to the current file to read file data read. To the buffer, if all path file read failure, switching path failure, directly to the end of the file read, above is the whole process of reading multi path file.

The principle of multi path read: in order to guarantee the broadcast system broadcast process, the data source effectively and continuously, need to be able to read the file path. All of the broadcast audio files stored in a file server and save the storage path [4], when the network fault or the server cannot return data or file error, it can immediately identify the fault and switch to a valid data source next path diagram, as shown in Figure 2.

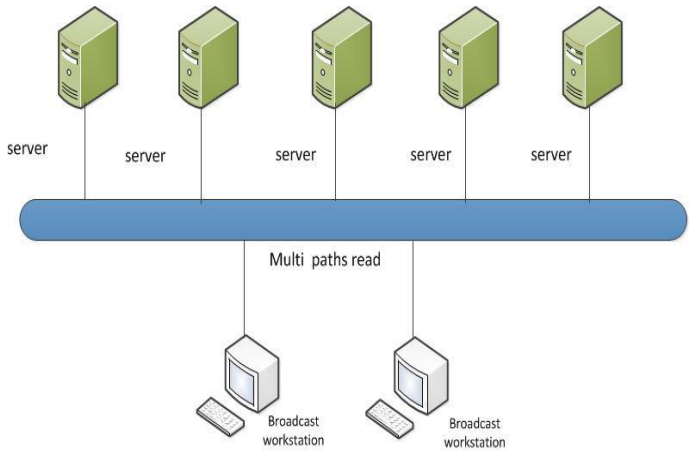

Figure 2: Multi Path File Read Schematic Diagram

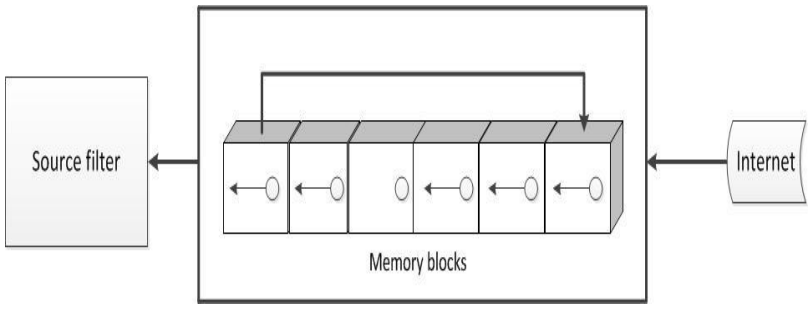

Figure 3: Buffer Queue Structure Diagram

The principle of data buffer mechanism: Our data buffer actually is a circular queue. Queue is a kind of first in first out (first in first out, abbreviated as FIFO) of the linear table data structure, it only allows to insert one end of the table, and the other end to remove elements, in the queue, allowing the insertion of the tail end is called the rear, allowing the deletion end become the head of line (front) [5]. Source Filter to read data from the network from the buffer queue, write data is completed, returned to the buffer at the end of the queue, waiting to write data, through the operation of front pointers and the rear pointers, so reciprocating cycle using buffer queue data. The buffer queue structure diagram as shown in Fig. 3.

Secondly, introduces the principle of Source Filter, Filter is present in the form of COM component, and it is responsible for receiving data. The DirectShow platform provides a lot of filter to complete a variety of receiving, decoding, playback and other functions. The issue of the Source Filter is not only to complete the data receiving but

also need to read and cache data path, namely the buffer cycle queue and multi path reading technology in the Source package in Filter, do the receiving side buffer.

3. Design and Implementation of Files Seamless Switch Technology Based On Direct Show

This section will describe the implementation of this method in DirectShow tools. Design and implementation of the module is divided into two processes, the first is the design and implementation of a data buffer, followed by the design and implementation of multi path to read Source Filter.

3.1 Design and Implementation of Data Buffer

The main function of the data buffer is writing the file data into data buffer for broadcast use according to the capacity of the buffer and the data buffer filling strategy before or during the digital audio workstation system broadcasts. Even if network interruption failure or the file itself data error occur, it also won't make a play to stop immediately. It provides time for security to the backup file path for switching, so as to ensure the continuous non-stop broadcasting system.

According to the principle of the data buffer mechanism, the data buffer is designed as a circular queue. When broadcasts in the process, it reads audio data sequentially from the circular queue and then completes the block of memory to read data, back to the circulating at the end of the queue, waiting for the next filling so that read and fill cycling.

Due to the design of the data buffer to set and adjust its capacity, and the buffer size is closely linked to the audio format, we first introduce the audio broadcast audio format S48. S48 (stereo, 48kHz) by using MPEG-1 layer 1, MPEG-1 layer 2 (Mp1, Mp2) audio compression format, due to its easy to edit and shear, the S48 format has become a standard format for broadcast audio workstation and automatic broadcasting station which has excellent sound quality and meet the requirements broadcast audio editing precision. The buffer size is equal to the product of rate and time. According to the S48 format bit rate, or rate, the audio data we need in the buffer time can be calculated, but we need to set the buffer size. If the buffer size is K, S48 format code rate is 256kb/s, the audio data buffer time is K * 8=256 * t.

The data buffer capacity adjustment strategy: In order to avoid buffer loading too much data, K data buffer size is set to 1.5 and the data buffer capacity is 1.5M and the audio data is equivalent to setting the buffer 48 seconds before the broadcast system broadcast. When the broadcast system in the process of data broadcast, the appropriate buffer adjustment but the highest not to exceed 10M (audio data size of 320 seconds), in order to avoid buffer overload, resulting in reduced efficiency workstation.

The data buffer filling strategy: After broadcasting system began to broadcast, if the buffer data amount is less than the buffer threshold R (i.e. the minimum buffer filling value), start writing data process, filling the buffer data. The data buffer is filled by the capacity constraint strategy adjustment strategy. The data buffer K value and R threshold can be set. The data buffer filling process flow diagram as shown in Fig.4.

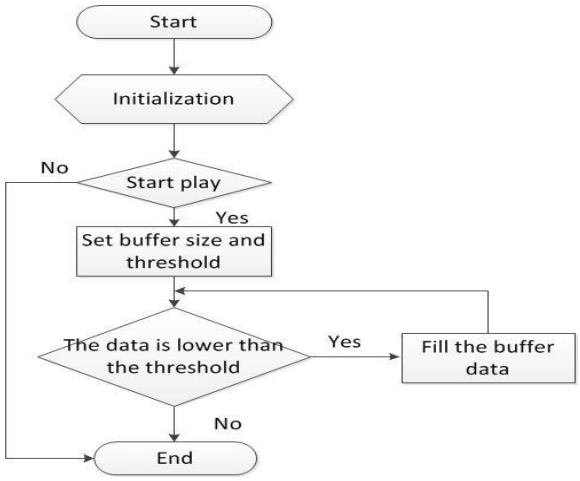

Figure 4: The Flow of Filling the Data Buffer

The data buffer filling process as follows: firstly, to initialize the data buffer, setting the data buffer size of K is 1.5, the threshold R was 0, and reads the file data filling the buffer; secondly, to determine whether to start playing, if not start playing, then filled the data buffer completely; and then, if you start playing, setting the buffer size of K is 10, the threshold R is 1.5; thirdly, whether the data buffer data is less than the threshold, if less than the threshold R, continue to fill the buffer or the end of the data buffer filling.

The CBufferFiller class was supplement to the CFileStream that supported for the CFileBufferStream's functions and accomplished the process of the buffer.

Its structure diagram was shown as Figure 5:

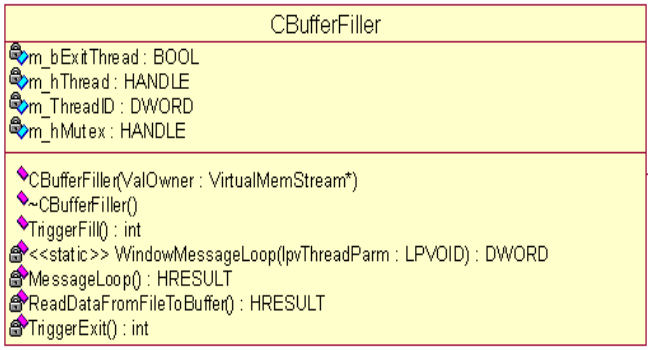

Figure 5: Structure Diagram of Cbufferfiller Class

The key functions in the CBufferFiller were described as bellow:

- CBufferFiller(VirtualMemStream* ValOwner)

Function: Constructor, initialize the accept message process;

Return: null;

Parameter1:ValOwner；VirtualMemStream* type; virtual files data stream.

- int TriggerFill()

Function: Trigger filling data;

Return: int; if the return value was zero, function exceptions successfully, others were fault code;

Parameters: null.

- HRESULT ReadDataFromFileToBuffer()

Function: Read the files data into the buffer;

Return: HRESULT; return function exceptions successfully or not;

Parameters: null.

In this class, a message mechanism is used and it triggered the "TrriggerFill" message by the messages in the queue, then called the "ReadDataFromFileToBuffer" function to fill the data.

3.2 Design and Implementation of File Multi-Path Reader Module

The multi-path reader module for a file, is designed for broadcast system data source interrupted which may be caused by network failure or audio file error. In order to achieve the reader for multi-path, it must sure that one audio file may be stored in several path which may be in a computer or at different computers or in a disk matrix, and all the path need to be saved in database. When error occurred, the play position of a file will be recorded and the source file will be switched into a valid one, after that read new file data from the same position, which can make the data source continuous and effective.

CFileStream class is a decomposition as class CFileBufferStream function, the main function is to select the file path , reads the file from the web server and records have been read the file size.

The class diagram as shown in Fig.6:

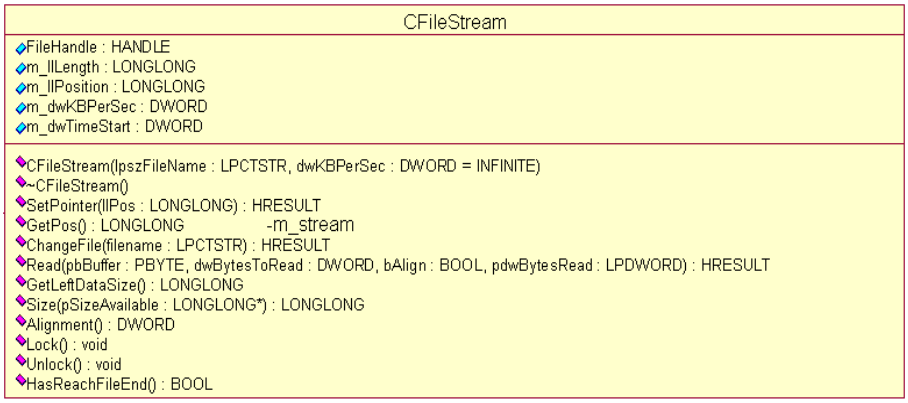

Figure 6: The Cfilestream Class Structure Diagram

The key function in the CFileStream:

- CFileStream(LPCTSTR lpszFileName, DWORD dwKBPerSec = INFINITE)

Function: Instantiate CFileStream class and initialized;

Return: null;

Parameter 1：lpszFileName；LPCTSTR type; will be read the file name string pointer;

Parameter 2：dwKBPerSec; DWORD type; will be read the audio file rate;

- HRESULT SetPointer(LONGLONG llPos)

Function: Set the file access location;

Return: HRESULT; return function exceptions successfully or not;

Parameter 1: llPos; LONGLONG type; file access location;

- HRESULT ChangeFile(LPCTSTR filename)

Function: Modify the current file;

Return: HRESULT; return function exceptions successfully or not;

Parameter 1: filename; LPCTSTR type; to change the file name.

- HRESULT Read(PBYTE pbBuffer,
  DWORD dwBytesToRead,
  BOOL bAlign,
  LPDWORD pdwBytesRead)

Function: Reads the specified length from the data in the file;
Return: HRESULT; return function exceptions successfully or not;
Parameter 1: pbBuffer; PBYTE; Buffer pointer;
Parameter 2: dwBytesToRead; DWORD type; Read data size;
Parameter 3: bAlign; BOOL type;
Parameter 4: pdwBytesRead; LPDWORD type; Read the file data, move the pointer position.

Through initialize this CFileStream class instance, loading the file and setting file access location (SetPointer), then start reading the specified length files into the buffer. If network interruption or file read and write errors, call ChangeFile to replace the current file, switch to the backup path, reading from the original files' interruption point.

3.3 Design and Implementation of Multi Path to Read Source Filter

3.3.1 Function Analysis

Overall demand for multi path to read Source Filter can read and buffering audio file data from the network source when the network failure disconnected or error audio file data, the ability to switch to an alternate path, continue to read the file data and the file buffer from the point of interruption.

1) Function single

In principle, Filter should be a single function module. If functions are too complex, you should be decomposed into multiple filters to achieve. In general, no matter what the function of Filter is the separation or integration, you should be adhere to the "moderate" principle to decide based on the actual situation. Not in the pursuit of a single function, the function is divided carefully, we design multi path to read the Source Filter, the need for a certain amount of output in the Pin data cache. In this case, there is no need to separate the realization of a Filter to implement caching. Therefore, it needs only a single filter to implement all the function of the above conditions.

2) Choose one filter module

Multi path read audio file data Filter is mainly responsible for data acquisitions, data source is an audio file data in a file server, and then the data to transmission, so the filter should be Source Filter, responsible for data acquisition and transmission [6] down.

3) The definition of the input and output

It requires an output pin to read source filter by multi path.

4) The definition of the interface

Considering the previous Source Filter contains no multipath read function, so it is necessary to define the interface of IMultiFileSourceFilter multi path file.

5) The other special requirement

As a digital audio workstation broadcasted the underlying Filter system, it has the property of commodity, so the need for commercial property protection on the filter. The concrete realization method is in the process of establishing filter, check whether there is a dog, if not, then exit the program, so as to ensure the safety of the property rights system software.

3.3.2 The Design of the Filter

We can realize this Seamless Switch method in DirectShow Filter, which is simply described as follow.

Overall demand for multi path to read Source Filter can read and buffering audio file data from the network source when the network failure disconnected or error audio file data, the ability to switch to an alternate path, continue to read the file data and the file buffer from the point of interruption.

As software development, in order to better realize the function, good scalability and robustness, the development of the filter also need the process of the design. Filter design mainly includes the following two aspects: the choice of an appropriate superclass and structural design.

1) Choose an appropriate class

The development of a Filter, it is important to choose the right parent class. Filter is a process within the COM components. Filter development is the COM component. DirectShow SDK provides the massive base, the base class is the realization of the basic data communication framework and COM rules through detailed analysis and comparison to it, is helpful to read Source Filter multi path to choose an appropriate the parent class, which can greatly reduce the difficulty of development. CasyncFilter. this paper realizes the multi path to read Source Filter, the class hierarchy as shown in Fig.5.

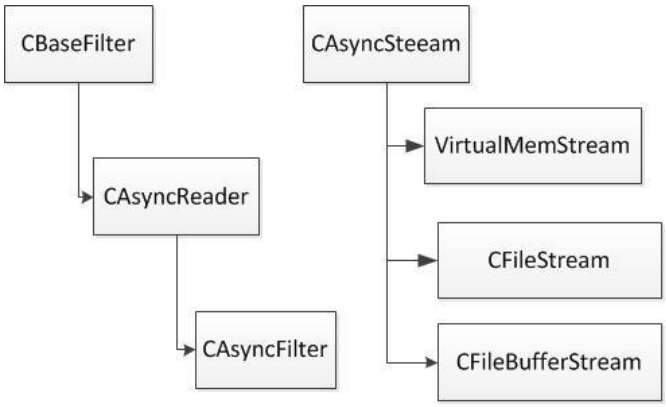

Figure 7: Source Filter Inheritance Graph

2) Application of structural design

Simply, Filter is just an application framework, one typical process which can support "input internal processing output". Under this framework, most important is application logic (commercial logic or control logic etc.), namely "internal process". So this paper would separate the design of the application framework and application logic design. And then the Source Filter structure would clearly and good expansibility. As shown in Fig.6.

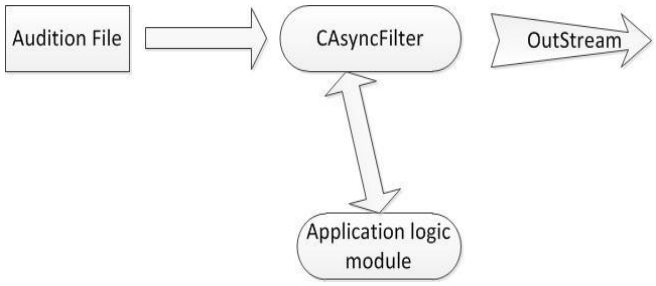

Figure 8: Filter Framework and the Application Logic

The application logic is that a control strategy for processing input data. As for data input and output the processed, these problems are in charge of by the Filter framework. What the application logic concerns more is to realize the efficiency of data processing, algorithm optimization. Application of logic control objects were always in the form of a composite member objects Filter.

3.3.3 The Realization of the Casyncfilter

In this paper, multi path buffer read the file Source filter class shown in Fig.9.

The CAsyncFilter is a Source Filter instance, the Source Filter inheritance graph shows the inheritance relationship, mainly realizes the asynchronous mode of CBaseFilter function. CAsyncFilter is a digital audio workstation broadcast core system for seamless switch implementation, implementation of the Filter framework to complete the output reading and audio streaming audio files.

The class diagram of CAsyncFilter was shown in Fig.10.

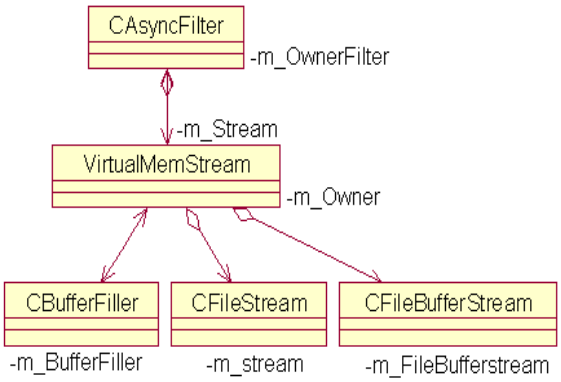

Figure 9: The Source Filter Function Graph

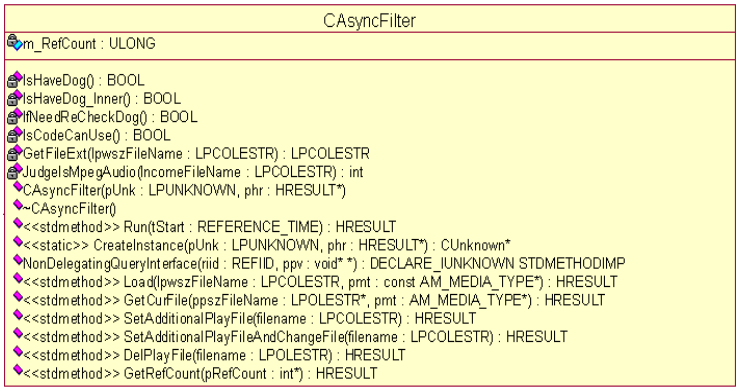

Figure 10: Casyncfilte Class Structure Diagram

The key function in the CAsyncFilter:

- CAsyncFilter(LPUNKNOWN pUnk, HRESULT *phr)

Function: Construct the CAsyncFilter's process.

Return: Null.

Parameter 1：pUnk; LPUNKNOWN type;

Parameter 2: phr; HRESULT *type;

CAsyncFilter inherits from CAsyncReader, the main function is to instantiate CasyncReader. At the same time to complete the necessary initialization operation, such as: the call IsHaveDog () function, to determine whether there is a dog, if not, then failed to initialize, exit the software.

- STDMETHODIMP Load(LPCOLESTR lpwszFileName，const AM_MEDIA_TYPE *pmt)

Function: loading a new file to VirtualMemStream queue;

Return: STDMETHODIMP, exceptions the return function successfully or not.

Parameter 1：LpwszFileName; LPCOLESTR type; the new file name string pointer;

Parameter 2：PMT; const AM_MEDIA_TYPE; the specified media type const pointer;

- STDMETHODIMP SetAdditionalPlayFileAndChangeFile(LPCOLESTR filename)

Function: setting up additional files to play and changing to the set switch;

Function return value: STDMETHODIMP, the function returns the successful execution;

Parameter1: Filename; LPCOLESTR type; additional files to plat full name string pointer;

- BOOL IsHaveDog()

Function: to detect whether there is encrypted dog;

Return: BOOL; return function implementation success;

Parameters: none;

In view of the present software environmental unsafe factor, in order to guarantee the security of system software, system in the implementation process of the underlying key Filter, joined the dongle detection to ensure the security by encrypting the dog technology copyright.

- int JudgeIsMpegAudio(LPCOLESTR IncomeFileName)

Function: to determine whether Mpeg audio file;

Function return value: Int; if it is Mpeg audio files, return 0, else if it is the Wave format, return 1, if it is the other format audio files, return 2, other numerical error code;

The parameter 1: IncomeFileName; LPCOLESTR format file name string pointer.

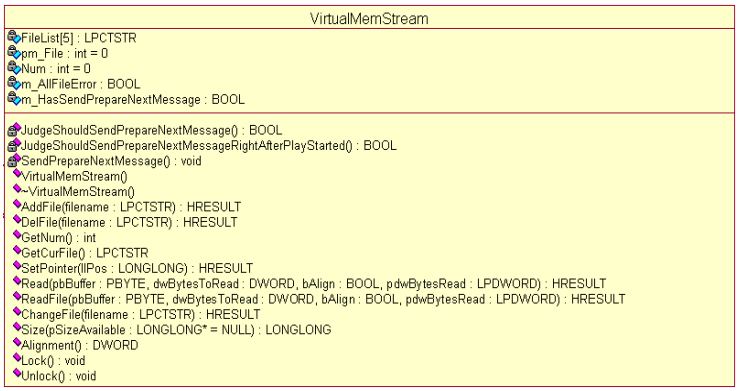

Figure 11: The Virtual Memstream Class

VirtualMemStream class is a virtual file data stream and form similar to a file type, so that it have the relevant attributes, such as: when the application calls the VirtualMemStream class, it is responsible provided the file data stream to the upper level program. At the same time, it also coordinated work by calling the CBufferFiller class, CFileStream class and CFileBufferStream classes, etc. to complete the multi path on the audio files to read and load the data into the buffer operation.

The class diagram as shown in Fig.11.

The key function in the VirtualMemStream:

- BOOL JudgeShouldSendPrepareNextMessage();

Function: to determine whether it should send to a program in the news;

Return: BOOL; return function implementation success;

Parameters: none;

- BOOL JudgeShouldSendPrepareNextMessageRightAfterPlayStarted()

Function: judge after the start should immediately prepare, only the remaining buffer is smaller than the 200KB material, it shall immediately prepare;

Return: BOOL; return function implementation success;

Parameters: none;

You can see the whole process by multi path buffer file read flow chart. When failed to read file, calling JudgeShouldSendPrepareNextMessage to determine whether read the end file, if the end is called SendPrepareNextMessage.

- HRESULT AddFile(LPCTSTR filename)

Function: to add a new file;

Return: HRESULT; return function implementation success;

The parameter 1: Filename; LPCTSTR type; the new file name string pointer;

- HRESULT DelFile(LPCTSTR filename)

Function: delete a backup file;

Return: HRESULT; return function implementation success;

Parameter 1: Filename; LPCTSTR type; remove the file name string pointer;

VirtualMemStream is a file stream, adding files by AddFile, DelFile deletes files so that realize the multi path on file read. Multi path file calls AddFile to add the backup file path, when the error file or network outages, calling the DelFile delete the file, switch to the standby file path and then continue to read the file.

- HRESULT ReadFile(PBYTE pbBuffer,

DWORD dwBytesToRead,
BOOL bAlign,
LPDWORD pdwBytesRead);

Function: read a piece of data;

Return: HRESULT; return function implementation success;

Parameter 1: pbBuffer; PBYTE type; the buffer pointer;

Parameter 2: dwBytesToRead; DWORD; to read data length;

Parameter 3: bAlign; BOOL type;

Parameter 4: pdwBytesRead; LPDWORD; to read data pointer;

- HRESULT ChangeFile(LPCTSTR filename)

Function: to switch files；

Return: HRESULT; return function implementation success;

Parameter 1: filename; LPCTSTR; switching file name string pointer.

- HRESULT SetPointer(LONGLONG llPos)

Function: perform find operations

Return: HRESULT; return function implementation success;

Parameter 1: llPos; LONGLONG type;

The main function of CFileBufferStream class is to file buffer data stream, which is responsible for reading the file from the network path caching and also records the read position, as well as the buffer space control, reasonable and timely to read documents and buffer, so as to avoid because the file data due to insufficient play out interrupt.

The class diagram as shown in Fig.12.

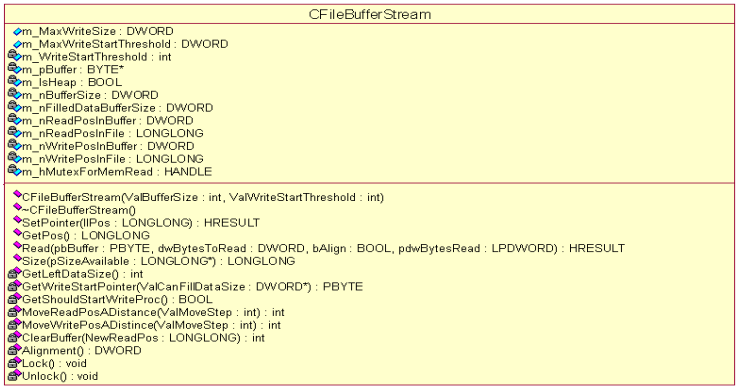

Figure 12: Cfilebufferstream Class

The key function in the CFileBufferStream class:

- LONGLONG GetPos()

Function: gets the current file position

Return: LONGLONG; read position returns the current file (relative to the file header)

Parameters: none;

- HRESULT Read(PBYTE pbBuffer,
  DWORD dwBytesToRead,
  BOOL bAlign,
  LPDWORD pdwBytesRead)

Function: reads the specified number from the data in the buffer;

Return: HRESULT; return function implementation success;

Parameter 1: pbBuffer; PBYTE; buffer pointer;

Parameter 2: dwBytesToRead; DWORD; read data size;

Parameter 3: bAlign; BOOL type;

Parameter 4: pdwBytesRead; LPDWORD; read the data, the ReadPos mobile location pointer

- PBYTE GetWriteStartPointer(DWORD *ValCanFillDataSize)

Function: can write data to the head pointer;

Return: PBYTE; can write data to the head pointer;

Parameter 1: ValCanFillDataSize; DWORD * type; data space size can be written;

- BOOL GetShouldStartWriteProc()

Function: whether should start writing process;

Return: BOOL; whether we should start writing process, TRUE promoter, FALSE does not start;

Parameters: none;

In the broadcast before the start, in order to avoid the data buffer data loading too much, set the data buffer size 1.5M, when the start after the broadcast, it allows to increase the buffer size t. Read the file data to the buffer as described below, call GetWriteStartPointer to obtain data write head pointer, mobile write data MoveWritePosADistince backward; call GetShouldStartWriteProc to judge, the buffer data is less than or equal to the set threshold value, if yes, then start writing process [7]; if the writing process, network or file read and write error remember, the current file reading and writing position, switch to the standby network path, obtain the original file read and write position by calling GetPos, continues from the backup file when the front position to read file data.

4. Conclusion

After the function test of this method, we can find that this filter base on this method can make the broadcasting more coherence. seamless switch technology can ensure the seamless switching digital audio workstations broadcast safety requirements on current broadcast file, when the current broadcast file is missing, or the current broadcast file on the file server network interruption, or the current audio file on the file server error.

Acknowledgements

This work was supported by the ChunHui project of the Ministry of Education named "Research of speech emotion recognition technology using in mobile communication", which project number is z2014054..

References

[1] Honggang Cai. Research and implementation of Windows pure software dual hot backup system based on the dissertation. Nankai University (**2008**)
[2] Siu Lee. Data backup and recovery scheme for CATV machine room. Chinese cable TV (**2010**)
[3] Shengwei Guo. Research and implementation of virtual studio system based on DirectShow technology. The thesis. Southwest Jiao Tong University(**2007**)
[4] Hao Xu. Nonlinear editing network system in Gansu science and technology (**2004**)
[5] Yumin Yan. Data structure (C language version). Tsinghua University press.(**1997**)
[6] Wei Zhou DirectSound playback technology programmer (**2001**)
[7] Jufeng Cui. The automatic broadcast system broadcast information (**1998**)
[8] Wang Ming. Visual C + + serial communication technology and engineering practice of computer and network (**2003**)

Authors

Yi Guo (1982.11-), Now is working in School of Electrical and Information of Xihua University, and working as University Lecturer. Got the Doctor's degree in June 2013 at School of Automation of University of Electronic Science and technology in the field of Detection technology and automatic equipment. And now major in Pattern recognition and intelligent system, Artificial intelligence, Speech recognition, music transcription, radio and television technology.

Qiong Li (1990.11-), Now is studying in School of Automation of University of Electronic Science and technology in the field of Pattern recognition and intelligent system as a graduate student who is in the second year.